Tugas Besar IF2211 Strategi Algoritma

Penerapan String Matching dan Regular Expression dalam DNA Pattern Matching

Semester II Tahun 2021/2022

Disusun oleh:

Muhammad Naufal Satriandana 13520068

Vincent Ho 13520093

Farrel Ahmad 13520110

Institut Teknologi Bandung

Sekolah Teknik Elektro dan Informatika

DAFTAR ISI

Bab I

Deskripsi Tugas

Dalam tugas besar ini, anda diminta untuk membangun sebuah aplikasi DNA Pattern Matching. Dengan memanfaatkan algoritma String Matching dan Regular Expression yang telah anda pelajari di kelas IF2211 Strategi Algoritma, anda diharapkan dapat membangun sebuah aplikasi interaktif untuk mendeteksi apakah seorang pasien mempunyai penyakit genetik tertentu. Hasil prediksi tersebut dapat disimpan pada basis data untuk kemudian dapat ditampilkan berdasarkan query pencarian.

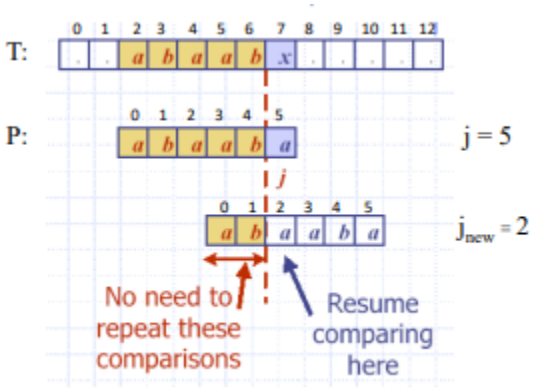
Bab II

Landasan Teori

2.1. Algoritma KMP

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma yang mencari pola di sebuah teks dengan cara kiri ke kanan, seperti halnya algoritma brute force. Akan tetapi, algoritma KMP melakukannya dengan lebih cerdas.

Jika terjadi mismatch antara teks dan pola P pada P[j], berapa penggeseran terbanyak yang bisa dilakukan untuk menghindari perbandingan yang sia-sia? Jawabannya adalah prefiks terbesar P[0…j-1] yang merupakan suffix P[1…j-1]



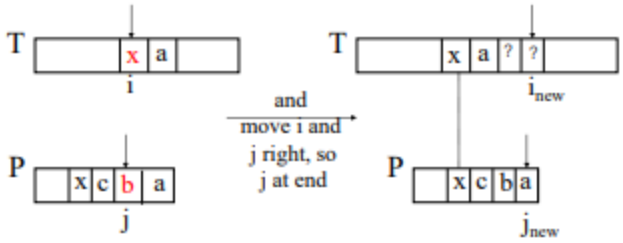
Hal tersebut dapat dilakukan dengan fungsi pinggiran b(k) yang didefinisikan sebagai ukuran prefiks P[0..k] terbesar yang juga merupakan sufiks P[1..k]. Kompleksitas waktu fungsi pinggiran adalah O(m) dan pencarian string adalah O(n), jadi kompleksitas waktunya adalah O(m+n), lebih cepat dibanding brute force.

2.2. Algoritma BM

Algoritmaa Booyer Moore adalah algoritma pencocokan string yang berdasarkan dua teknik, yakni looking glass technique (dengan mencari P di T dengan cara bergerak mundur) dan character jump technique. Character jump technique adalah ketika mismatch terjadi di T[i]==x, karakter di pola P[j] tidak sama dengan T[i], lalu menimbulkan 3 kasus:

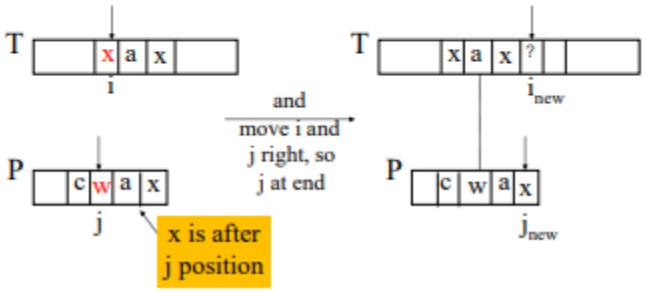
Case 1

➢If P contains x somewhere, then try to *shift P* right to align the last occurrence of x in P with T[i].



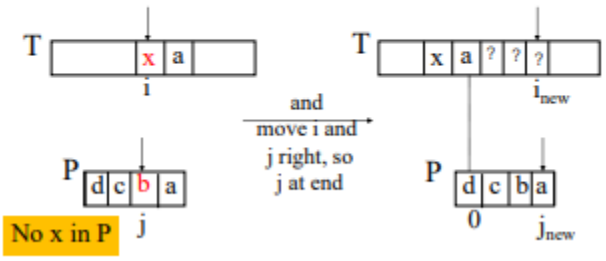
Case 2

➢If P contains x somewhere, but a shift right to the last occurrence is *not* possible, then *shift P* right by 1 character to T[i+1].



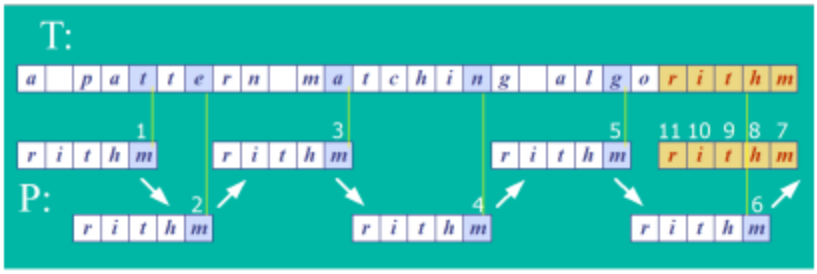
Case 3

➢If cases 1 and 2 do not apply, then *shift* P to align P[0] with T[i+1].



Berikut adalah contoh algoritma Booyer-Moore

Boyer-Moore Example (1)



Jumlah perbandingan karakter: 11 kali

2.3. Regex

Regular expression (ekspresi reguler) adalah sebuah baris karakter yang mendefinisikan pola pencarian, terutama digunakan di pattern matching. Berikut adalah pola regex dasar:

brackets [] : disjunction

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

Brackets [] ditambah garis sambung: range

RE	Match	Example Patterns Matched
/[A-Z]/	an uppercase letter	"we should call it 'Drenched Blossoms'"
/[a-z]/	a lowercase letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

- caret ^ : negasi

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	"Oyfn pripetchik"
[^Ss]	neither 'S' nor 's'	"I have no exquisite reason for't"
[^\.]	not a period	"our resident Djinn"
[e^]	either 'e' or '^'	"look up ^ now"
a^b	the pattern 'a^b'	"look up a^b now"

- Tanda tanya ? : bisa ada bisa tidak

RE	Match	Example Patterns Matched
woodchucks?	woodchuck or woodchucks	"woodchuck"
colou?r	color or colour	"colour"

- Titik: . any character

RE	Match	Example Patterns
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

2.4. Penjelasan Aplikasi Web

2.4.1 Frontend

Program ini menggunakan React sebagai framework atau library javascript untuk frontend. Karena frontend menggunakan React, aplikasi nampak hanya satu halaman atau single page aplication (SPA), yang mempercepat waktu loading. Pada frontend, tidak ada algoritma apapun selain untuk render halaman dan komunikasi ke backend. Komunikasi ke backend mengunakan Axios, yakni promise based http client. Jika frontend ingin mengambil data dari backend, maka dilakukan Axios.get. Jika ingin memasukan data ke DBMS, maka akan menggunakan Axios.post

2.4.2 Backend

Program ini menggunakan NodeJS dengan ExpressJS sebagai backend. Kegiatan olah data dari dan ke DBMS diatur melalui aplikasi backend NodeJS yang dibuat. Untuk DBMS, digunakan mysql dengan server buatan secara online yang dapat diakses kapanpun dan dimanapun. DBMS ini dihubungkan dengan aplikasi backend menggunakan package mysql pada NodeJS. Selain itu backend juga menjadi tempat regex, algoritma KMP, dan algoritma BM dieksekusi.

Komunikasi backend dengan frontend menggunakan GET dan POST pada REST API. Komunikasi dan data yang dikirim antar keduanya menggunakan struktur data JSON (JavaScript Object Notation). Ketika backend menerima request GET dari frontend pada endpoint tertentu, maka backend akan mengambil data dari DBMS dan dikirim kembali sebagai response ke frontend. Implementasi GET adalah pada riwayat penyakit untuk mendapatkan daftar riwayat penyakit dari DBMS. Sama halnya dengan POST pada endpoint tertentu, akan tetapi POST digunakan apabila request adalah data yang akan masuk ke DBMS. Implementasi POST adalah pada input penyakit dan tes dna, hal ini dikarenakan keduanya akan memasukkan data ke DBMS melalui backend. Apabila terdapat error seperti input sequence tidak valid, input nama penyakit sudah ada, ataupun lainnya maka akan dikembalikan sebuah response dengan status 400 yang artinya *bad request* dan dikembalikan juga pesan error yang lebih spesifik.

Bab III

Analisis Pemecahan Masalah

Langkah-langkah yang dilakukan untuk memecahkan masalah adalah sebagai berikut.

a) Menambahkan penyakit baru
   Hal pertama yang harus dilakukan sebelum melakukan pengujian DNA adalah menambahkan sequence DNA dari penyakit yang ingin diuji. Sequence DNA akan diterima dalam bentuk file text (.txt) dan kemudian dilakukan sanitasi input menggunakan regex (hanya boleh terdiri dari character ‘A’, ‘G’, ‘T’, ‘C’). Program kemudian akan mengecek apakah nama penyakit dan sequence DNA dari penyakit tersebut sudah ada sebelumnya pada database. Jika sudah terdapat pada database, maka akan diberikan pesan error yang berisi bahwa penyakit atau sequence DNA sudah terdaftar. Jika belum terdaftar, maka penyakit dan sequence DNA-nya akan ditambahkan ke dalam database.

b) Melakukan tes DNA
   Setelah memiliki daftar penyakit, maka langkah selanjutnya adalah memasukkan informasi dari pengguna yang akan diuji. Informasi yang perlu dimasukkan adalah nama pengguna, sequence DNA pengguna, dan nama penyakit yang ingin dicocokan. Nama pengguna dan nama penyakit akan divalidasi agar tidak kosong, jika kosong maka program akan menampilkan pesan error. Kemudian nama penyakit akan dicari pada database. Jika penyakit tidak ditemukan, maka akan ditampilkan pesan error. Sequence DNA pengguna juga akan dilakukan sanitasi input menggunakan regex (hanya boleh terdiri dari character ‘A’, ‘G’, ‘T’, ‘C’). Pengetesan DNA akan melakukan string matching dari sequence DNA penyakit terhadap sequence DNA pengguna. Algoritma KMP dan Boyer-Moore akan digunakan untuk melakukan pengujian tersebut, apabila hasil dari algoritma KMP dan Boyer-Moore menghasilkan true (ditemukan substring sequence DNA penyakit pada sequence DNA pengguna) maka hasil tes adalah true dengan tingkat similarity sebesar 100%. Selain itu akan mengeluarkan false.

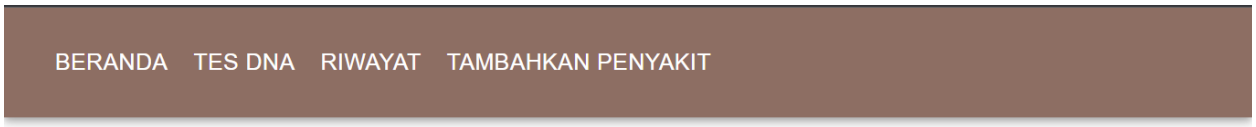
c) Melihat History / riwayat tes DNA
   Untuk melihat riwayat penyakit, terdapat search bar yang dapat digunakan untuk filter riwayat penyakit. Apabila search bar tidak diisi maka akan menampilkan seluruh riwayat penyakit dari database. Apabila search bar berisi tanggal saja maka akan menampilkan riwayat penyakit

berdasarkan tanggal input. Sama halnya dengan nama penyakit saja maka akan menampilkan riwayat dengan nama penyakit yang sesuai input. Apabila diisi keduanya, yaitu tanggal dan nama penyakit maka akan menampilkan riwayat penyakit yang sesuai dengan input tanggal dan nama penyakit.

Bab IV

Implementasi dan Pengujian

Tampilan menu utama

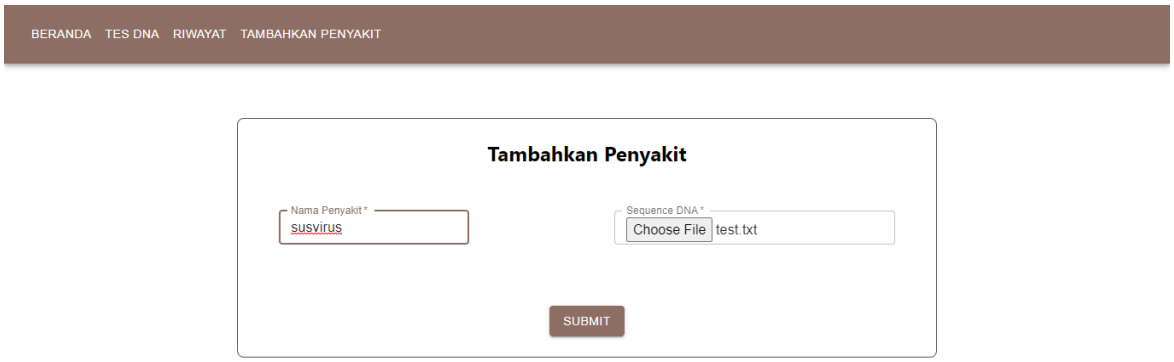


DNA PATTERN MATCHING

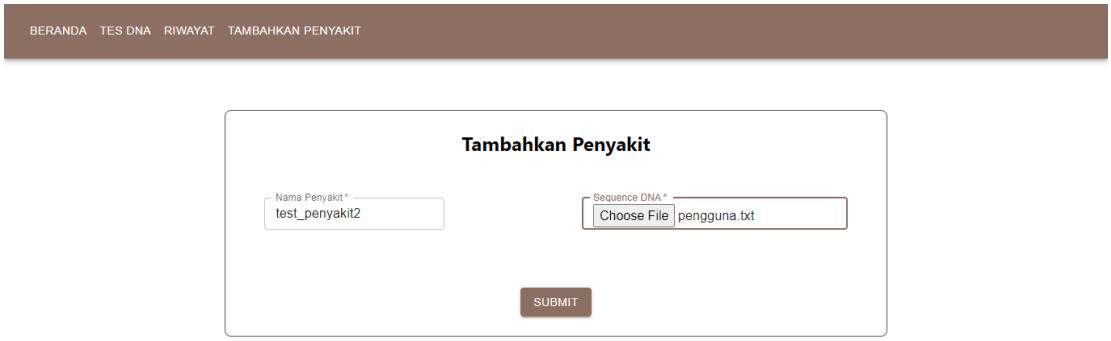
Developed by:

Muhammad Naufal Satriandana 13520068
Vincent Ho 13520093
Farrel Ahmad 13520110

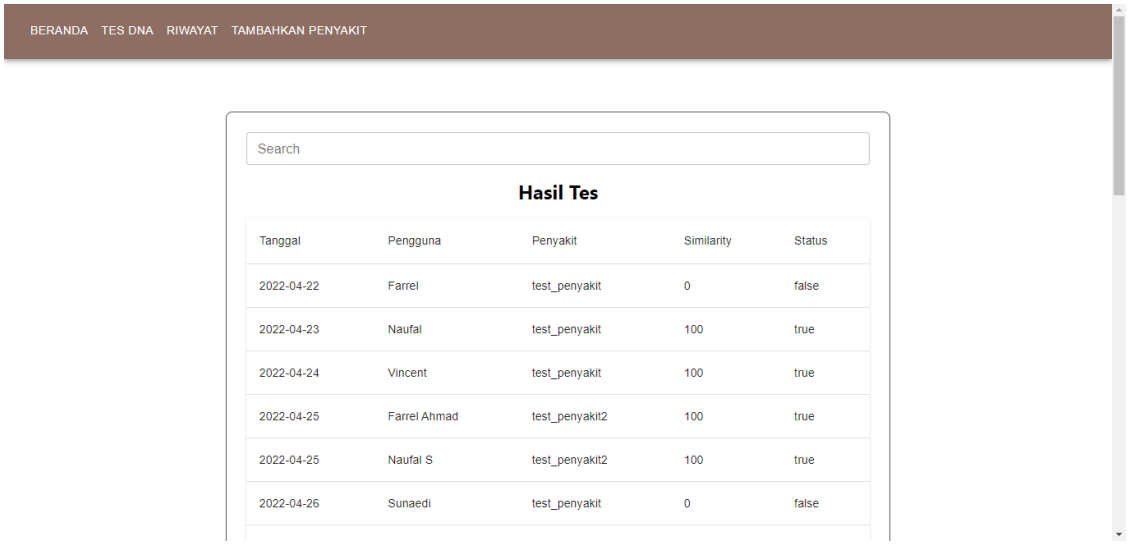
Tampilan menu tambahkan penyakit



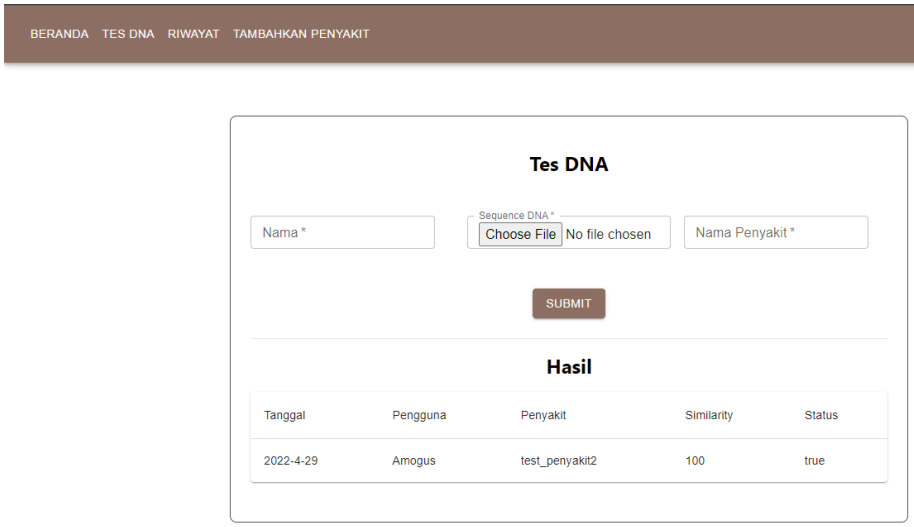
Tampilan menu tes dna



Tampilan menu riwayat hasil tes



Memasukkan tes DNA
(similarity tidak diimplementasikan, hanya tampilan saja disiapkan sehingga hasil similarity adalah diskrit 0 jika false dan 100 jika true)



Penggunaan search bar pada riwayat tes DNA

Bab V

Kesimpulan dan Saran

5.1 Kesimpulan

Algoritma KMP dan Boyer Moore dalam string matching lebih cepat dibanding algoritma brute force, sehingga sangat cocok untuk digunakan dalam pencocokan DNA.

5.2 Saran

A. Frontend bisa dibuat lebih rapi dan lebih bagus lagi
B. Seharusnya, tampilan error message pada frontend tidak menggunakan alert karena dapat mengganggu, seharusnya diintegrasikan ke dalam tampilan
C. Bonus seharusnya bisa diimplementasikan
D. Koneksi DBMS pada backend sering mengalami putus koneksi akibat timeout. Seharusnya dapat dibuat *handle disconnect* untuk secara otomatis menghubungkan backend dengan DBMS kembali.

Daftar Pustaka

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf diakses 29/4/2022

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf diakses 29/4/2022

Links

Repo Github : https://github.com/vincen-tho/Tubes3_13520068